Week 1-Foundations of CNN

15 June 2020 10:29

Convolutions Neural Networks

Computer Vision

- Object detection.
- Image classification
- Neural Style transfer.

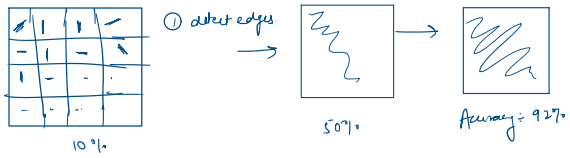
64×64 :- (64×64)×3
{r,g,b}

1024×1024 (Imgs) :- (1024×1024)×3
≈ 3 million.

(Very high)

Edge Detection.

Computer vision problems



1. detect Vertical lines. (|)
2. detect horizontal lines. (—)

- Vertical edge detection

* Shift up 1 column after iteration & row after iterating 1 row

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Image.

(6×6)×1

"Convolution"

1	0	-1
1	0	-1
1	0	-1

filter / kernel.

= 4×4 mat.

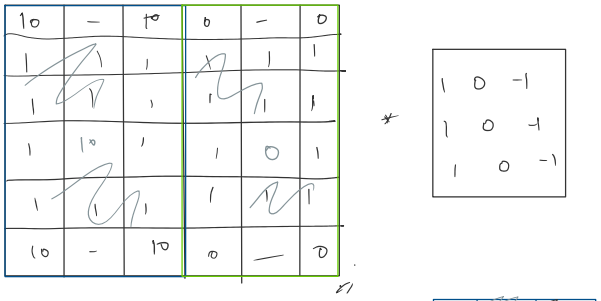
-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

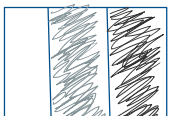
(3*1) + (0*0) + (1*-1) + (1*1) + (5*0) + (8*-1) + (2*1) + (7*0) + (2*-1).
= -5.

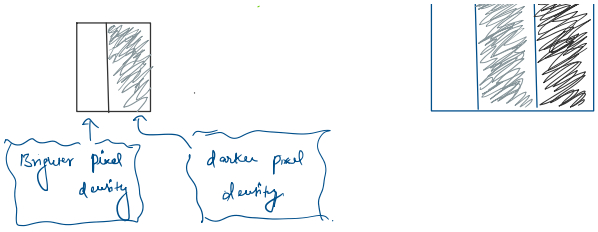
(0*1) + (1*0) + (2*-1) + (5*1) + (8*0) + (9*-1) + (7*1) + (2*0) + (5*-1) = -4

* (Image - Vertical Edge detector)

- tf. nn. Conv2d.







(i) pixel value $\propto$ $\frac{\text{bright pixels}}{\text{dark pixels}}$

Horizontal Edge Detection

1	1	1
0	0	0
-1	-1	-1

More filters

1	0	-1
2	0	-2
1	0	-1

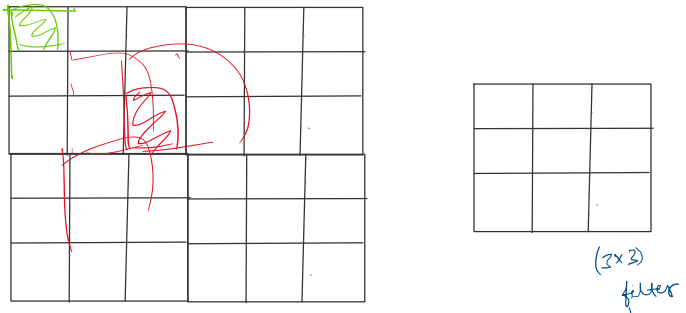
* Sobel filter

3	0	-3
10	0	-10
3	0	-3

* Scharr filter

Vertical Edge detection.

Padding



(6×6) Image

(3×3) filter

(n×n) (f×f)

$(n-f+1 \quad * \quad n-f+1)$

(6,6) $\xrightarrow[\text{shrink}]{}$ (4,4) ⇒ (1,1)

f(3,3).

- Image shrinking.
- Top left pixel (0,0) is used only once but centre pixel is overlapped many times.

- Top left pixel [0,0] is used only once but centre pixel is overlapped many times.
- We are loosing information from corner pixels.

* padding

$(6 \times 6) \rightleftharpoons (8 \times 8)$

$(8-3+1) = 6 \times 6$ image (previous image)

- padding of 0s

p: padding = 1 :- $(n-f+1)$

$n(+2p) - f + 1$

Valid and Same Convolutions

"valid" :- No padding.

$(n \times n) \rightarrow (f \times f) \Rightarrow (n-f+1) \times (n-f+1)$

"same" :- Pad so that input size is same as output size.

$(n+2p-f+1)$

$n + 2p - f + 1 = n$

$p = (f-1)/2$

* f is usually odd.

$f \in 2n+1$

- **Strided Convolution**

stride $\rightleftharpoons$ step

stride = 2 (say)

step by (stride)

padding: p
stride: s
image: n.

$\left[\left(\frac{n+2p-f}{s}\right)+1\right] \times \left[\frac{\quad}{\quad}\right]$

↳ if this is not integer then floor $(\mathbb{Z})$

Cross-Correlation v/s Convolution

① Sometimes we flip the filter (vertically and horizontally) and this process is called "Convolution" and process we perform is called Cross-correlation.

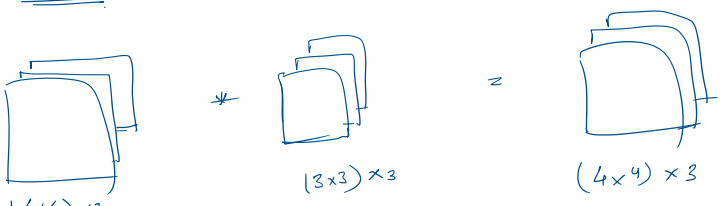
$$f \Rightarrow \begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 2 \\ -1 & 9 & 7 \end{bmatrix}$$

$$f* \Rightarrow \begin{bmatrix} 7 & 9 & -1 \\ 2 & 0 & 1 \\ 5 & 4 & 3 \end{bmatrix}$$

Convolution Over Volumes.

Convolution Over Volumes.

① On RGB



Height × width × channels

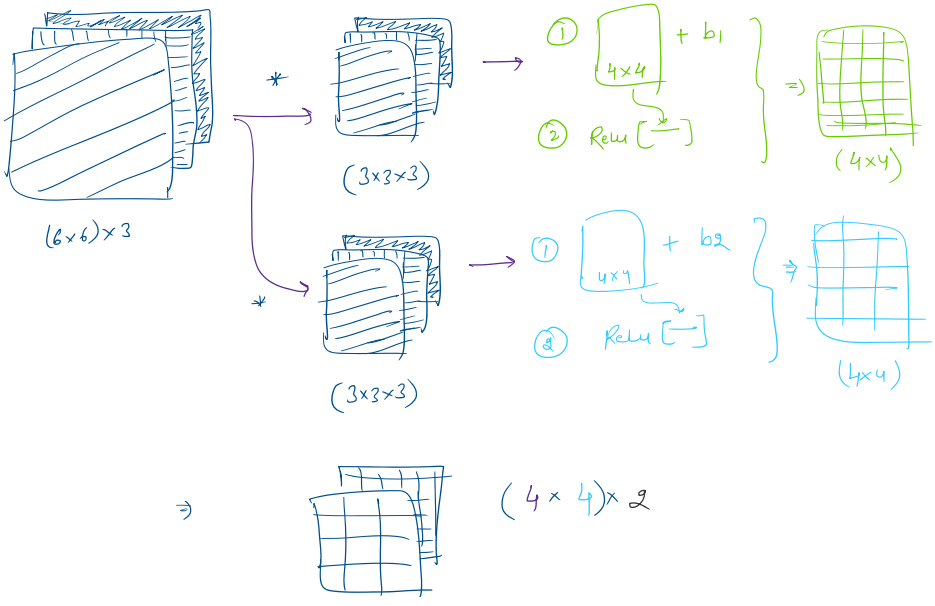
Correct * Channels of Image = Channels on filter.

$$(n \times n \times n_c) * (f \times f \times n_c) \Rightarrow (n-f+1) \times (n-f+1) \times n_c'$$

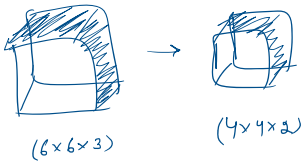
$(6\times6)\times3 \quad * \quad (3\times3)\times3 \qquad 4\times4\times2$

② Assuming stride = 1, padding = None.

One Layer of CNN.



$\Rightarrow$ $(4\times4)\times2$



1 Layer of ConvNet.

- If layer l is a Convolutional Layer :-

$f^{[l]}$ = filter size.

$p^{[l]}$ = padding

$s^{[l]}$ = Stride

$n_c^{[l]}$ = No. of filters.

Input : $(n_H^{(l-1)} \times n_W^{(l-1)} \times \boxed{n_c^{(l-1)}})$

O/P : $(n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]})$

$$n_{(H/W)}^{[l]} = \left[\frac{n^{(l-1)} + 2p^{(l)} - f^{(l)}}{s} + 1\right]$$

$f^{[l]}$ = filter size.

$p^{[l]}$ = padding

$s^{[l]}$ = Stride

$n_c^{[l]}$ = No. of filters.

Size of filter :>

$f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Input : $\left(n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]} \right)$

O/P :> $\left(n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]} \right)$

$n^{[l]}_{(H/w)} = \left[\frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s} + 1 \right]$

Activations :>

$a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

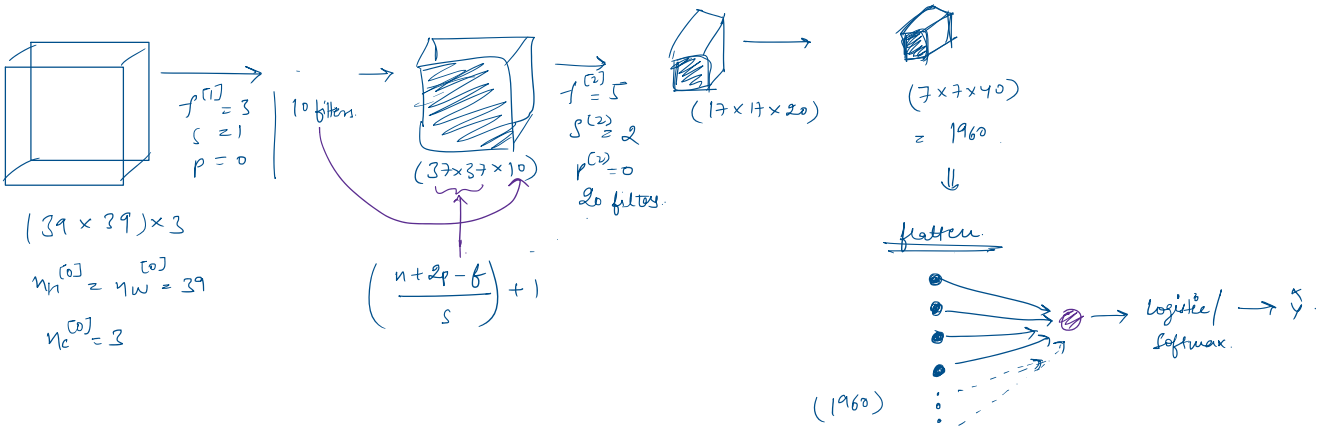
$A^{[l]} \rightarrow m \times a^{[l]}$

Weights :>

$\underbrace{\left(f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \right)}_{1 \text{ filter}} \times \underbrace{n_c^{[l]}}_{(\#\text{filters})}$

bias :> $n_c^{[l]}$ $(1 \times 1 \times 1 \times n_c^{[l]})$

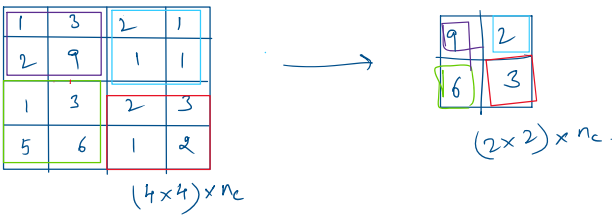
Simple Convolution N/w Example.



Types of N/w in CNN :>

- Convolution. (CONV)
- Pooling Layer (POOL)
- Fully Connected (FC)

Pooling Layers. : Max pooling



Hyperparameters ⇒

* filter, $f = 2$
* stride, $s = 2$

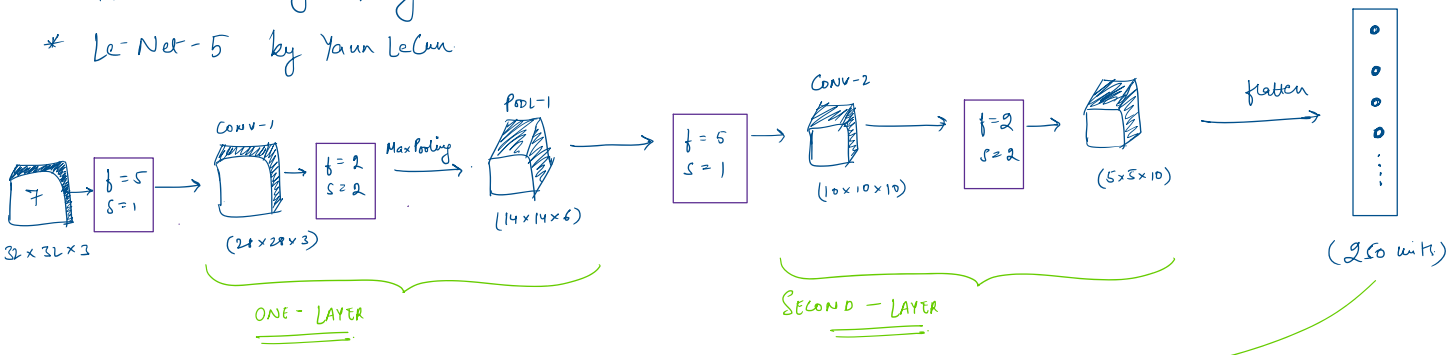
* Take max over 2×2 regions.
* Takes no parameters.

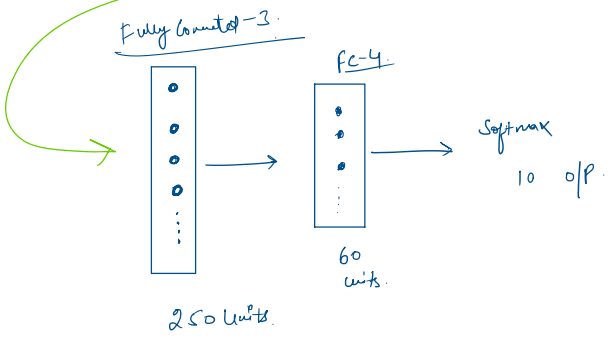
$$\left(\frac{n + 2p - f}{s}\right) + 1$$

- Less used : Average pooling.

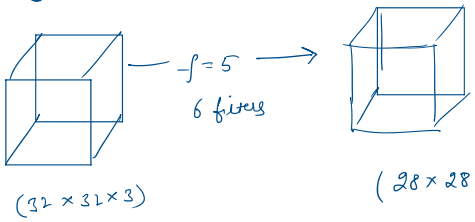
CNN Example

- Handwritten digit recognizer.
- Le-Net-5 by Yann LeCun



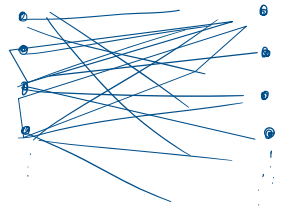


Why Convolutions. ?



$= 3072$

$= 4704$



3072×4704

$\approx$ 14 million.

#1 Parameter Sharing :-

A feature detector that's useful in one part of the image is probably useful in another part of image

#2 Sparsity of Connections :-

In each layer, each O/P value depends only on a small number of units.

1 O/P unit is connected with 6/36 units.

(n) (n×n)

Week 2- deep convolution models: Case Study?

18 June 2020 12:23

Classic N/w's

- Le- Net 5
- AlexNet
- VGG-16.

→ ResNet (Residual N/w).

152 layer nn.

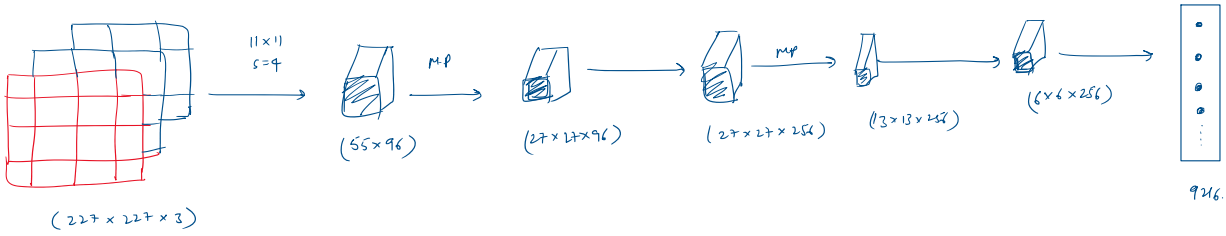
→ Inception.

- **Le Net - 5**

60k params.

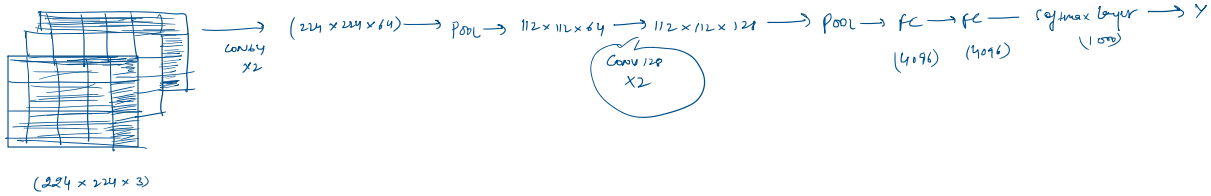
conv → pool → conv → pool → fc1 → fc2 → ŷ.

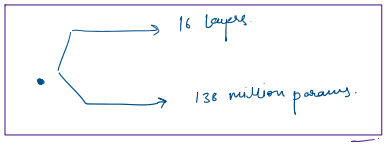
AlexNet



VGG - 16

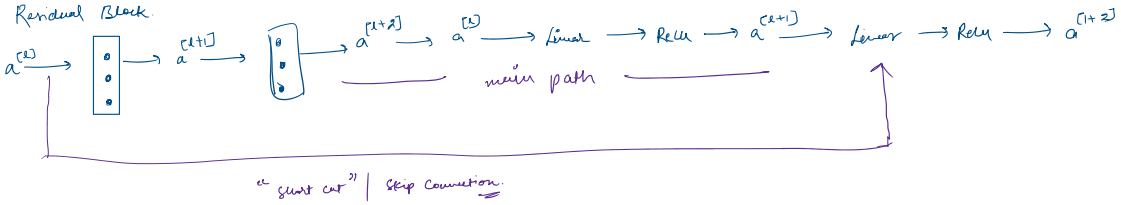
CONV = 3x3 filter, s=1, same.
MAX POOL = 2x2, s=2.

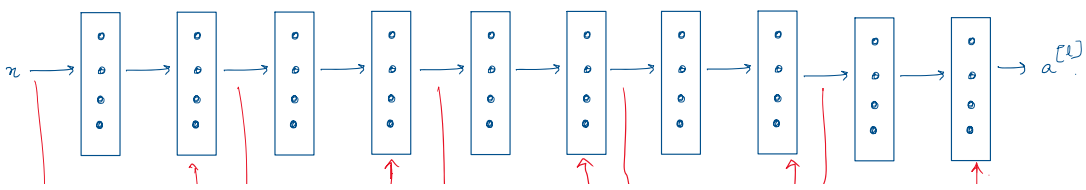


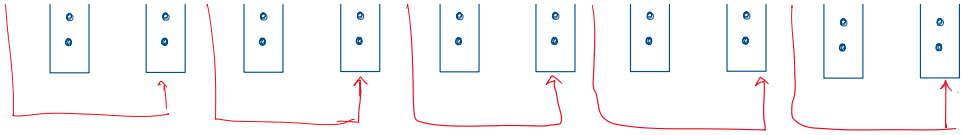


ResNets - Residual N/w's

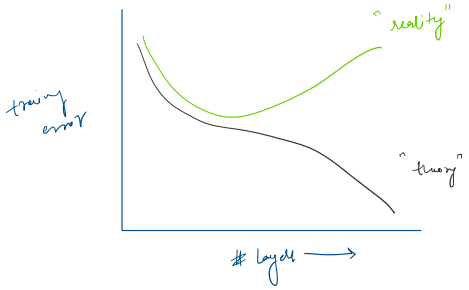
Residual Block.



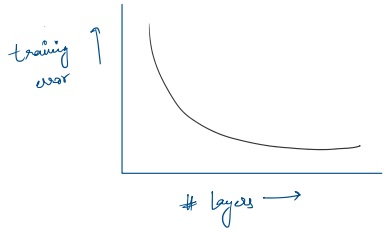




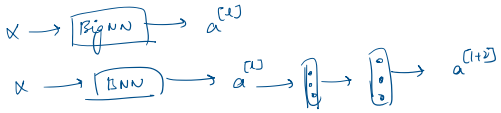
Plain.



ResNet



Why ResNets Work ?



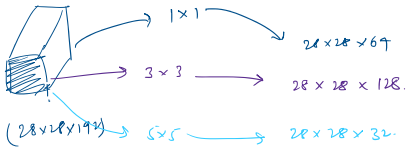
$\therefore$ (ResNet)

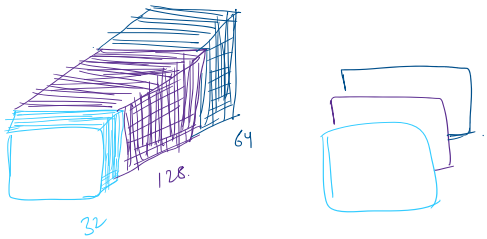
$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

$z = wa + b$

Inception N/W.

What to Choose?
Pooling layer / Conv. layer ?
No. of filters ?





(X) Computational Cost

(28 × 28 × 192) → CONV 5×5, same, 32 → (28 × 28 × 32)

$(28 \times 28 \times 32) \times (5 \times 5 \times 192)$

$\approx$ 120 million.

Reduce cost by 1×1 Convolutions !

- $(28 \times 28 \times 192)$ → CONV. 1×1, 16, 1×1×192 → $(28 \times 28 \times 16)$ → CONV, 5×5, 32, 5×5×16 → $(28 \times 28 \times 32)$

Also called "Bottleneck layer".

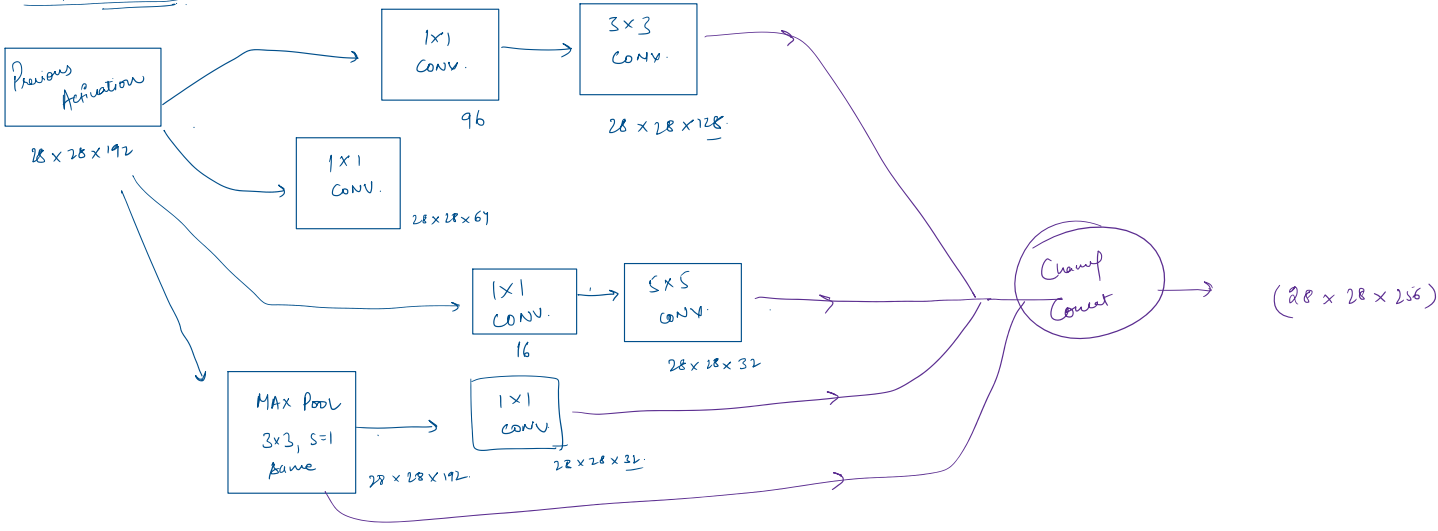
Convolutional Cost :→ $(28 \times 28 \times 16) \times 192 \approx 2.4$ million

+

$(28 \times 28 \times 32) \times (5 \times 5 \times 16) \approx 10$ million

$\approx$ 12.4 million

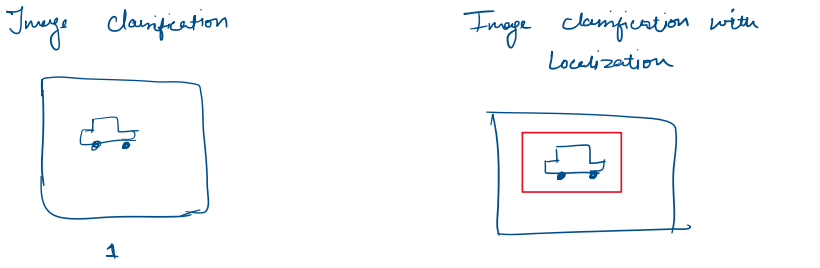
Inception N/W

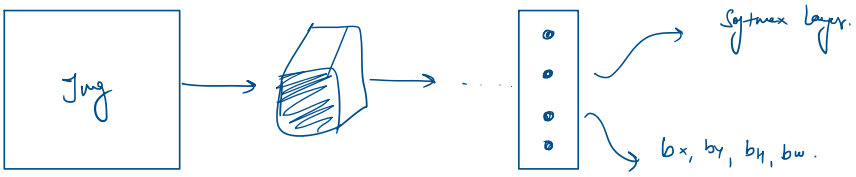


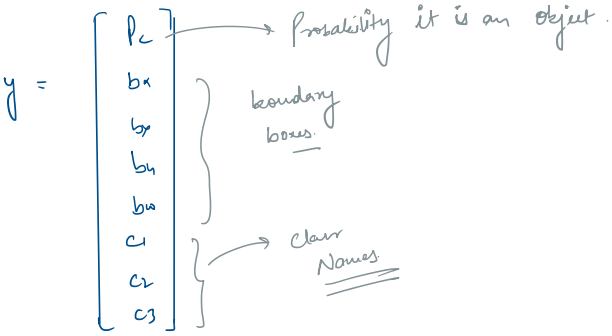
Week 3 - Detection Algorithms

19 June 2020 22:36

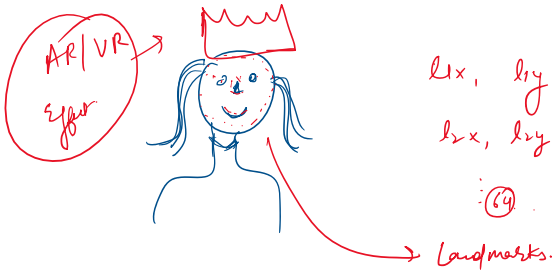
Object Localization





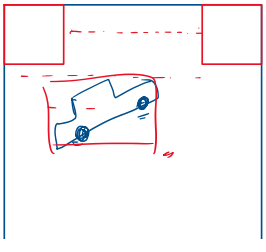


Landmark Detection



Object Detection

Sliding window detection



Traverse with whole window.

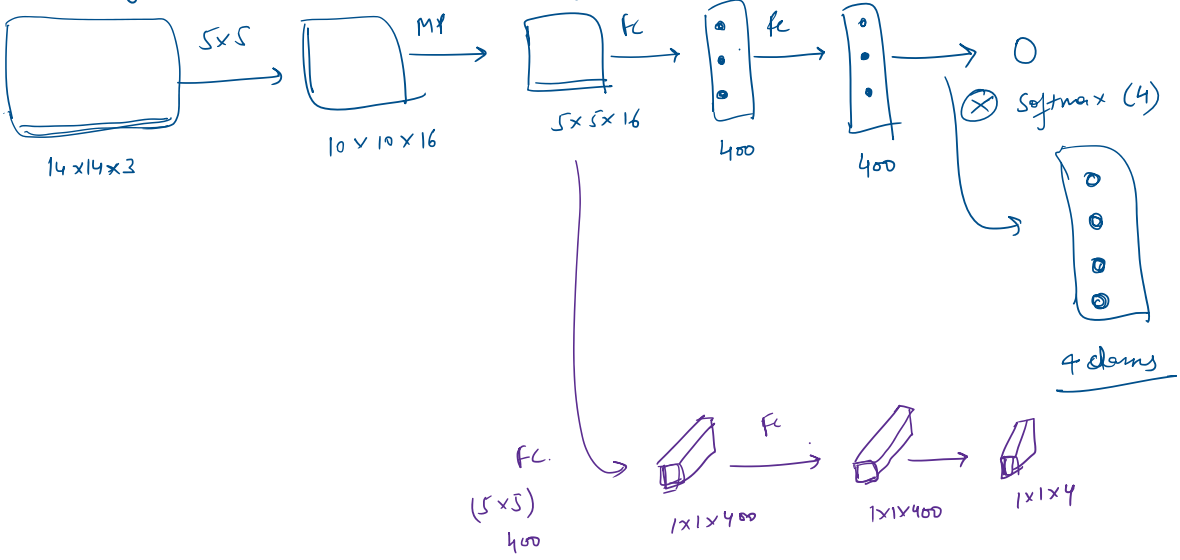
increase the window size.

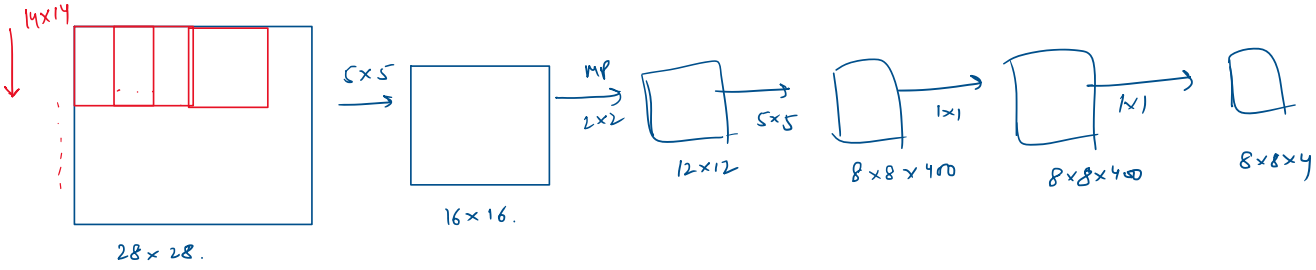
- Very Expensive Computational Cost

- Very Expensive Computational Cost

Convolution implementation of Sliding Window.

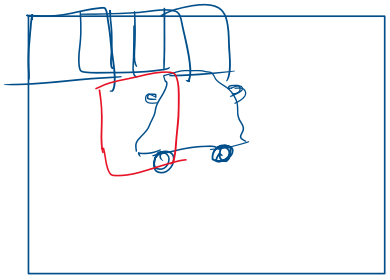
- Turning fc Layers into Convolutional Layers.





Bounding box prediction.

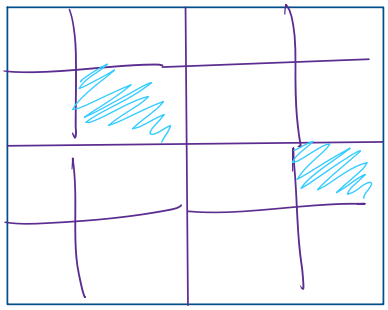
- Output accurate bounding boxes.



* None of the bounding box is Accurate!

YOLO Algorithm.

→ "You only look once"



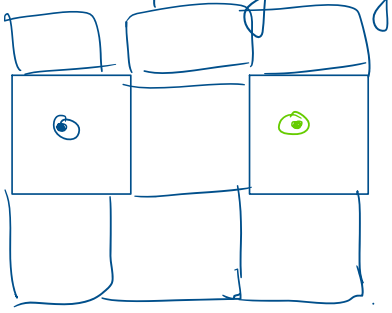
* divide into grids.
* Object to be detected.

Labels for training

for each grid cell -

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_H \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

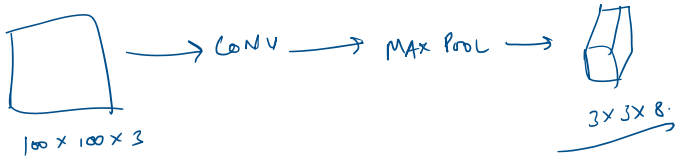
① Mid point of each grid cell is assigned if object is detected.



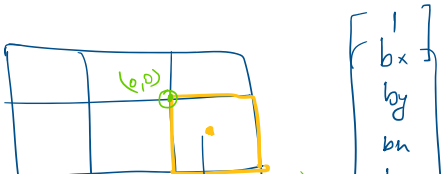
Total Volume of O/P :->

3 × 3 × 8

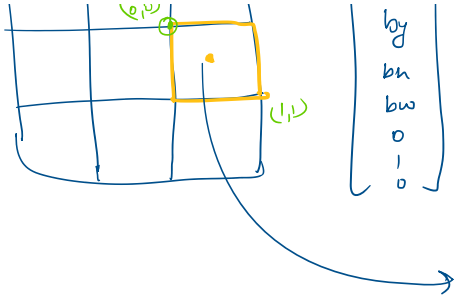
8 dimensional o/p vector.



Specifying the bounding box.



$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_H \end{bmatrix}$$



p_c
b_x
b_h
b_w
0
1
0

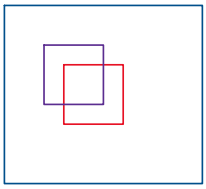
$b_x = 0.7$
$b_y = 0.3$
$b_w = 0.9$
$b_h = 0.5$

$0 < x < 1$

could be > 1

Intersection Over Union.

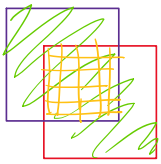
Evaluating Object Localization



□ : ground truth bounding box.

□ : Algorithm Output.

IOU



∪ Union

∩ Intersection

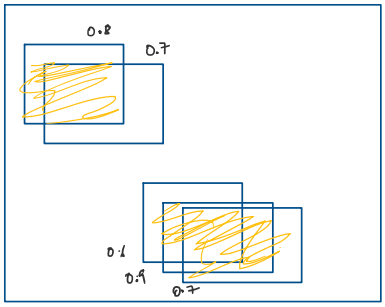
$$\frac{\text{Size of intersection } \cap}{\text{Size of union } \cup}$$

"Correct if IOU ≥ 0.6"

(*) IOU is the measure of Overlap.

Non-Max Suppression.

(x) Algorithm can detect same object multiple times, NMS makes sure that object is detected only once.

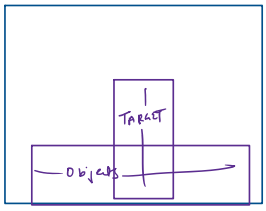


$p(c)$ = Probability of Object.

- Discard all boxes with $p(c) \leq 0.6$
  With the remaining boxes :→
- Pick the boxes with maximum $p(c)$.
- discard any remaining box with IOU ≥ 0.5 with the box output in the previous step.

ANCHOR BOXES.

(X) Only one object in the gridcell can be identified



predefine two (n) Anchor Boxes

- Anchor Box 1 
- Anchor Box 2 

$y =$

P_c
b_x
b_y
b_h
b_w
c_1
c_2
c_3
P_c
b_x
b_y
b_h
b_w
c_1
c_2
c_3

Anchor Box 1.

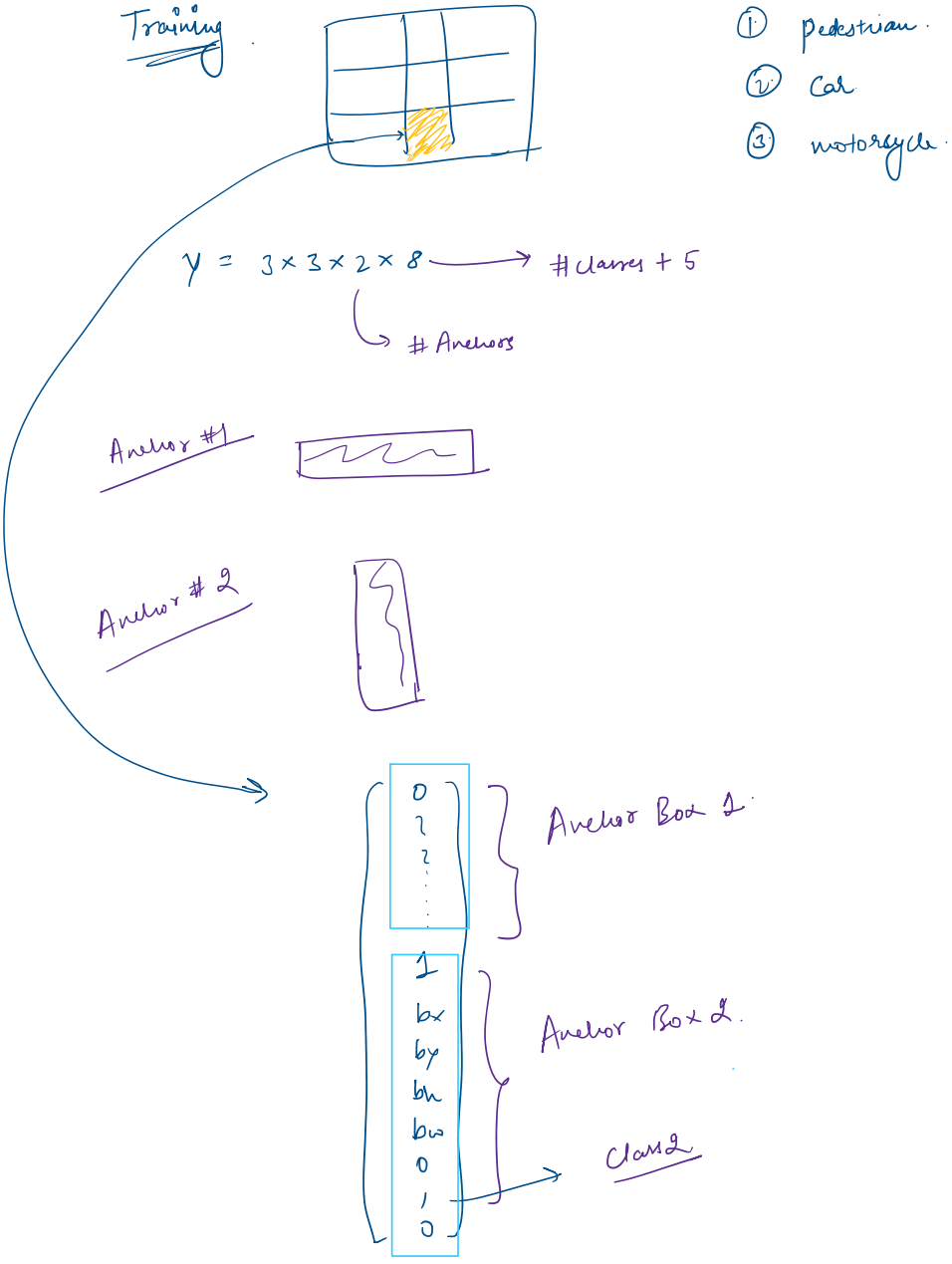
Anchor Box 2

$y = 3 \times 3 \times 16$
$\uparrow$
$(2 \times n)$
$(n=2)$

YOLO ALGORITHM : Putting it together.

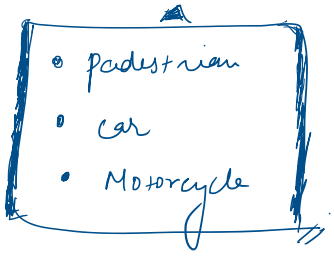
Training 

① pedestrian.



Outputting the non-max Suppressed O/P :→

- For each grid cell, get two predicted bounding boxes.
- Get rid of low probability predictions.
- For each class, use non-max Suppression to generate final predictions.



Week 4 - Special applications: Face recognition & Neural style transfer

21 June 2020 19:31

FACE RECOGNITION.

- Verification
  - → Input image.
  - → O/p whether the image is of claimed person (0/1).
- Recognition.
  - → Has database of K persons.
  - → get an input image.
  - → O/p Id of that person if in db.

ONE SHOT LEARNING.

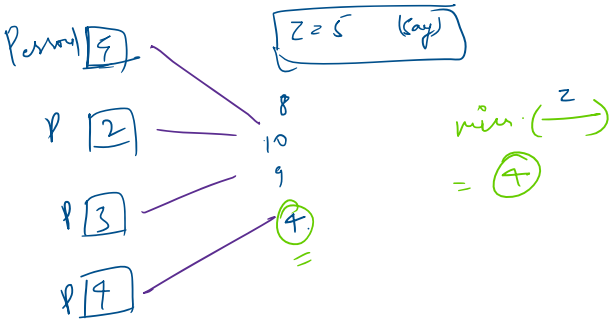
"Learn by only single input image".

* Learn similarity fx.

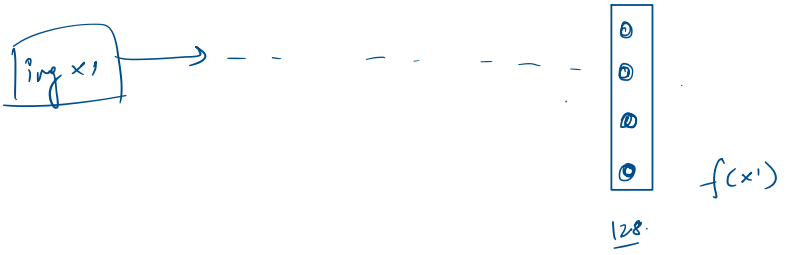
d(img1, img2) = degree of difference btw images img1 and img2.

if { d ≤ τ "Same" ; d > τ "different" }

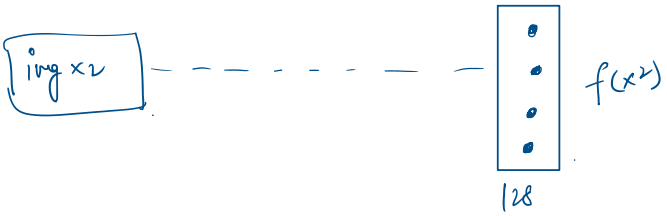
- τ = Tau (some hyperparameter)



SIAMESE NETWORK.



encoding of img(x1) = f(x1).

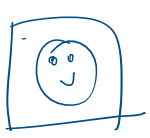


$$d(x_1, x_2) = \| f(x_1) - f(x_2) \|_2^2$$

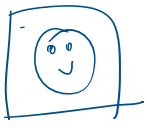
Comparing two images and calculating d is sometimes called "Siamese Network".

"Deep Face".

TRIPLET LOSS Fn.

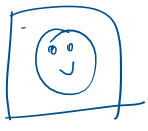


Anchor image.



Positive image.

$z \ll$



Anchor image.

Negative image.

$z \gg$

Triplet : { Anchor, A ; +ve, P ; -ve, N.

- $\underbrace{\| f(A) - f(P) \|^2}_{d(A,P)} \leq \underbrace{\| f(A) - f(N) \|^2}_{d(A,N)}$

$$d(A,P) - d(A,N) \leq 0$$

$$\| f(A) - f(P) \|^2 - \| f(A) - f(N) \|^2 \leq 0 - \alpha$$

(α : hyperparameter)

or

$$\left[\| f(A) - f(P) \|^2 - \| f(A) - f(N) \|^2 \right] + \underbrace{\alpha}_{\text{(Margin.)}} \leq 0$$

$\alpha = 0.2$ (say)

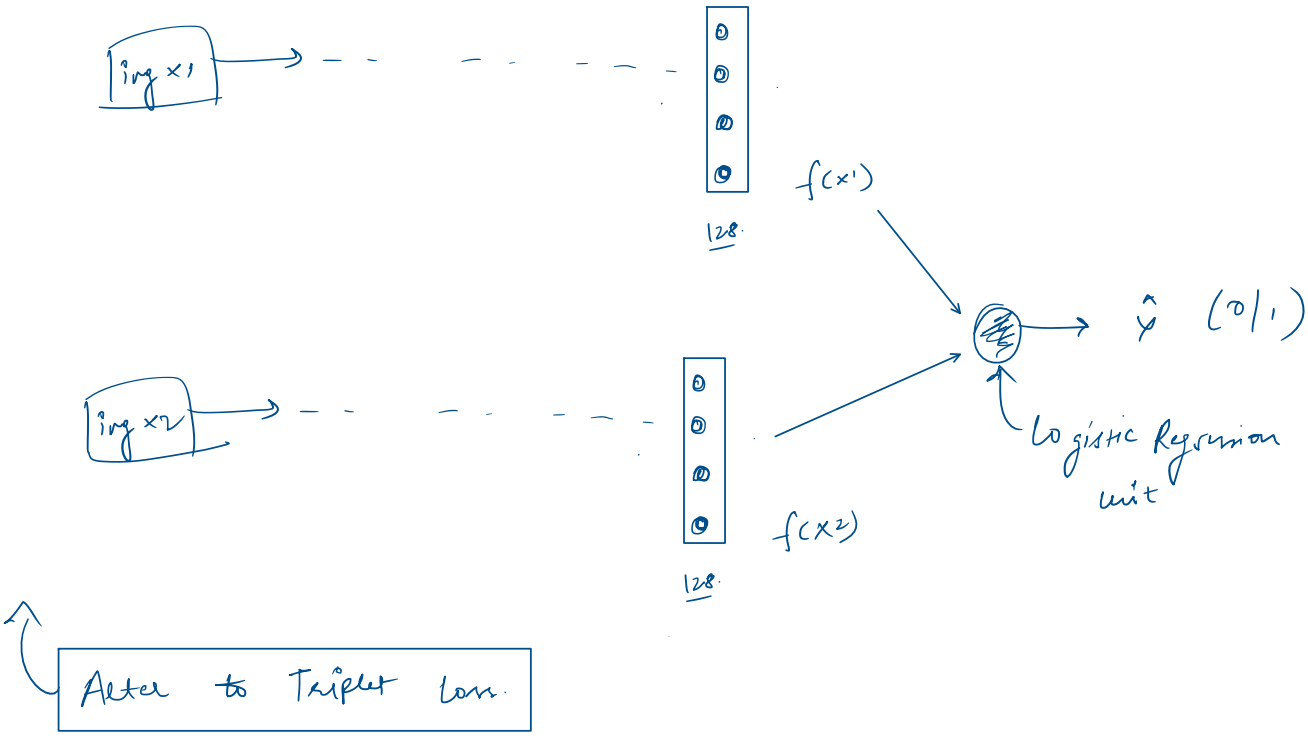
Loss fx.

- given three images A, P, N.

$$\mathcal{L}(A,P,N) = \max \left(\underbrace{\| f(A) - f(P) \|^2 - \| f(A) - f(N) \|^2 + \alpha}_{\leq 0}, 0 \right)$$

$$J = \sum_{i=1}^{m} \mathcal{L}(A_i, P_i, N_i)$$

Face Verification & binary Classification.



Alter to Triplet loss.

$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b\right)$$

NEURAL STYLE TRANSFER

Image 1 (Original) + Image 2 (filter Image) ⇒ Neural style transfer.

⊙ deep Convnets learning.

Pick a unit in Layer 1. Find the 9 image patches that maximize the units activation. Repeat for other units.

Cost fx.

Content C | Style S

→ generated image G

$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$

$J_{content}(C, G)$ → similarity.

Initiate G randomly,

G : 100 × 100 × 3

$G := G - \frac{\partial}{\partial G} J(G)$.

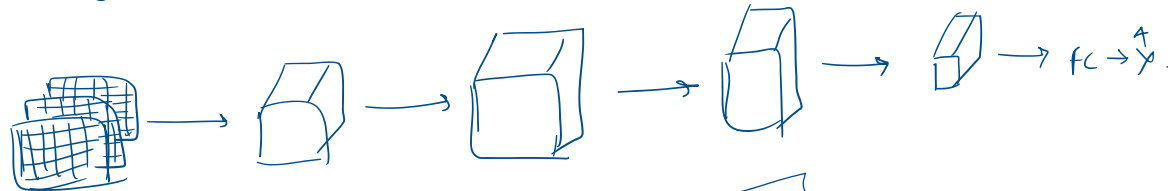
Content Cost fx.

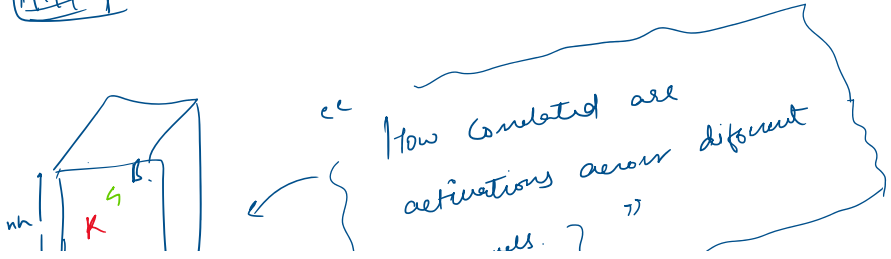
$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$.

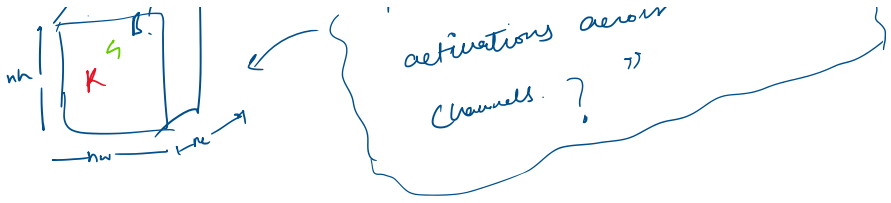
- Say you use layer l.
- Use pre-trained ConvNet (say VGG).
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be activation on layer l on the image.
- If $a^{[l](C)}$ & $a^{[l](G)}$ are similar, both images have similar content.

$$J_{content}(C, G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

Style Cost fx.







Style Matrix

Let $a_{ijk}^{[l]}$ = activation at (i, j, k)

H w c

$G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

$$G_{kk'}^{[l](G)} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} \left(a_{ijk}^{[l](G)} \cdot a_{ijk'}^{[l](G)} \right)$$

(Correlation)

$$G_{kk'}^{[l](S)} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} \left(a_{ijk}^{[l](S)} \cdot a_{ijk'}^{[l](S)} \right)$$

G = generated image

S = Style image

"gram matrix".

$$J_{style}^{[l]}(S, G) = \| G^{[l](S)} - G^{[l](G)} \|_F^2$$

$$= \frac{1}{(2 n_H^{[l]} n_W^{[l]} n_c^{[l]})^2} \sum_k \sum_{k'} \left(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)} \right)$$

$$J(G) = \alpha \left[J_{content}(C, G) \right] + \beta \left[J_{style}(S, G) \right]$$

1d and 3d generalization

2d image (14 x 14) * 2d filter (5 x 5) = (10 x 10)

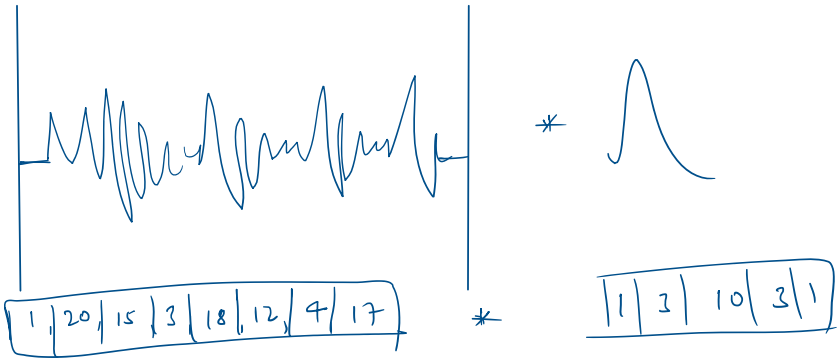
2d image * 2d filter = (10 × 10)

(14 × 14) (5 × 5)

Same can be applied to 1d image.

ex. ECG

electro cardio graphy.



1	20	15	3	18	12	4	17	*	1	3	10	3	1